Transformers, part 1

2021-01-12

Paper, Vaswani et al. 2017

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

iv:1706.03762v5 [cs.CL] 6 Dec 2017

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

https://arxiv.org/pdf/1706.03762.pdf

Architecture

Key features:

- **Attention**
- **better results (translation)**
- **More parallelizable**
- **Less train-time**
- **Generalization to new tasks**
- **‘simple architecture’**

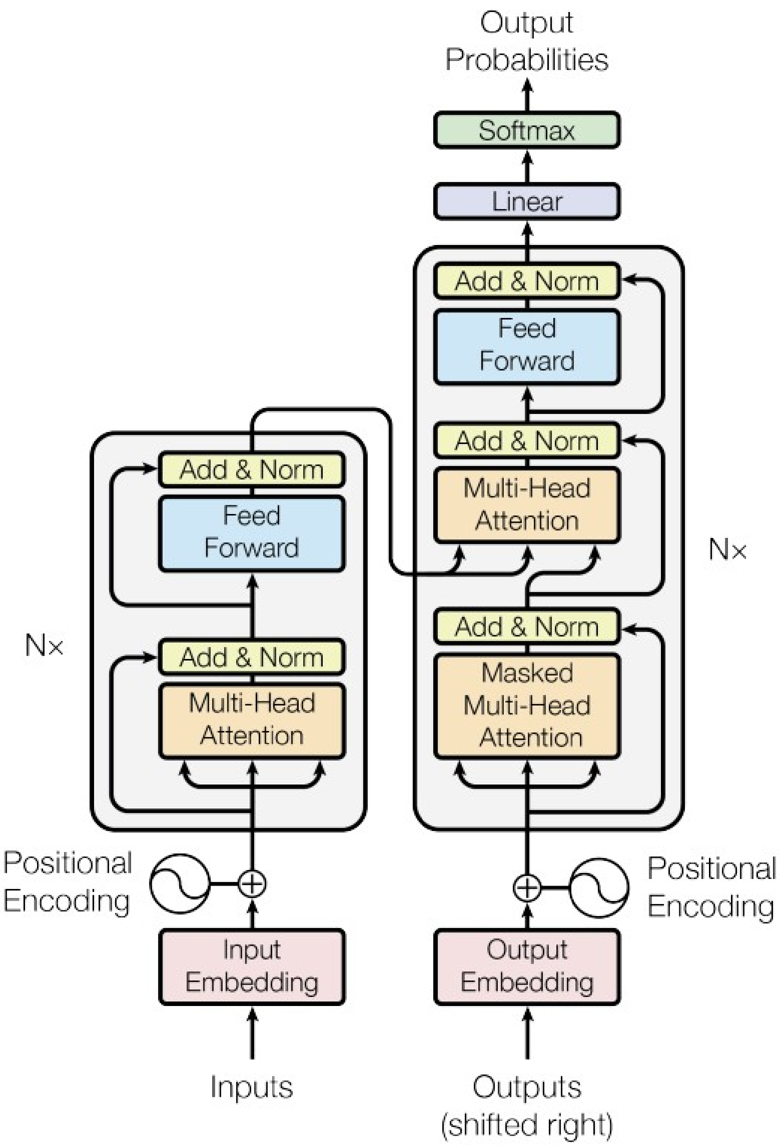


Figure 1: The Transformer - model architecture.

Transformers in the news

Generative capabilities

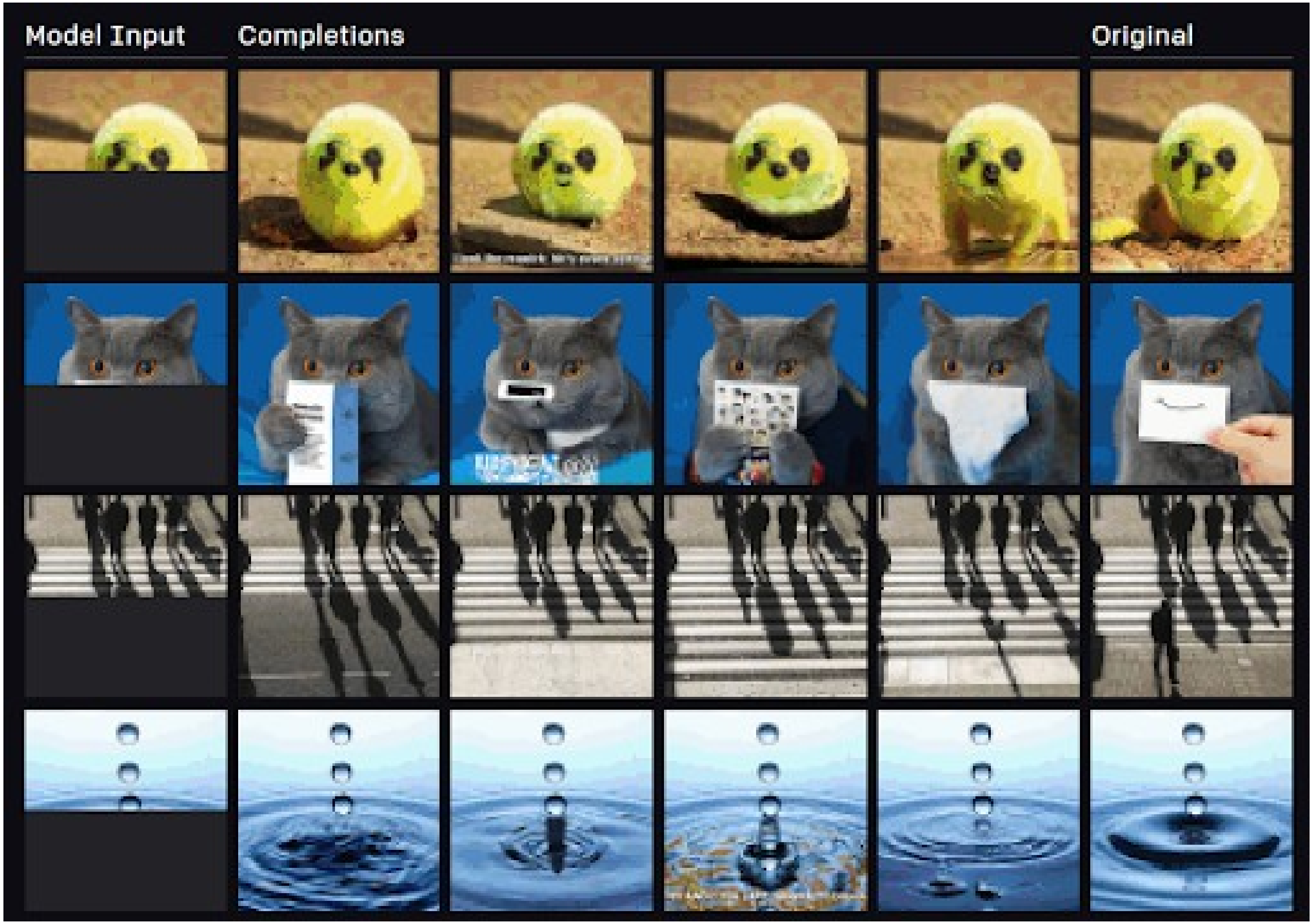
John St Clair Etouffee, world-renowned art expert, connoisseur, aesthete, art historian, and lecturer at the University of Oxford, is walking around the Louvre, savoring the exquisite artwork of all ages assembled before him. The Louvre is closed, allowing him free access to all the paintings and silence in which to meditate them. He stops in front of La Gioconda, Leonardo da Vinci's world-famous Mona Lisa. After a time, he begins to speak and describe the qualia, the frisson, the nigh-indescribable experience of the subtle swirls & *sfumato* of the painting:

"The sun is now barely a foot or two below the horizon and with every second, the golden tone of the painting deepens. Everything melts in it. I, who am having this experience, appear to you in a little corner of the painting and I look at you through the painting and our eyes meet. Our arms, our shoulders, our heads, our hands, our necks, our mouths, our lips, our eyes, our fingers and the tips of our toes—all this transforms itself into bright, beauteous colour. Everything becomes alive, all there is left in me is love and appreciation. It makes me want to shout at you "Don't leave!" but you don't hear me and I remain silent. La Gioconda looks at me and I at La Gioconda, the same as before, but now it's like being in two places at the same time and my feet have no feeling."

St. Clair Etouffee leaves the painting and enters the Hall of the Kings, and of Kings. He moves on to the Indian Wars, passes the Saloon d'Orléans, the King of Britain with his Tudor and Stuart wives, and is reminded of what St Clair had read a lifetime earlier at the age of twelve in a book about medieval paintings

https://mspoweruser.com/what-is-gpt-3-and-how-it-will-affect-your-current-job/

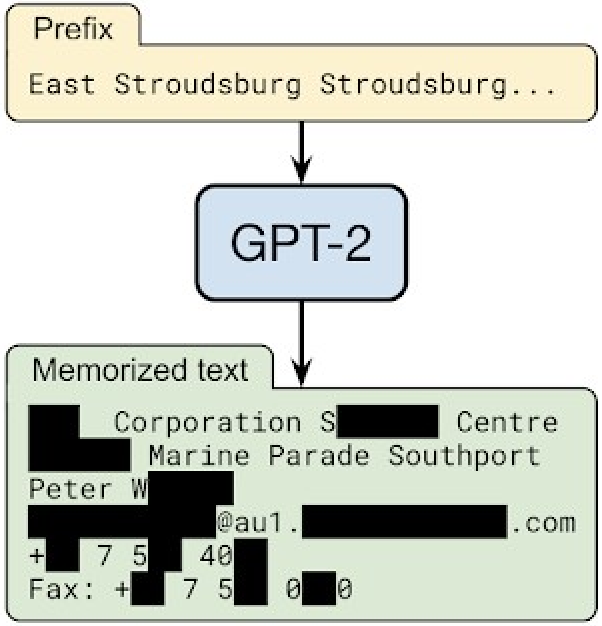
Transformers in the news



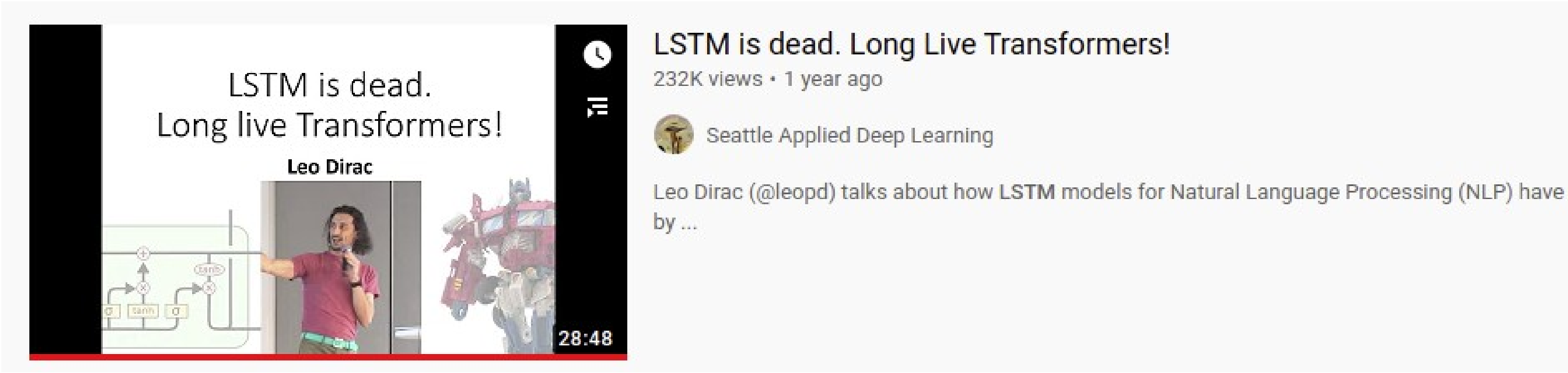
Privacy Considerations in Large Language Models

Trainend on large datasets that also contain sensitive data

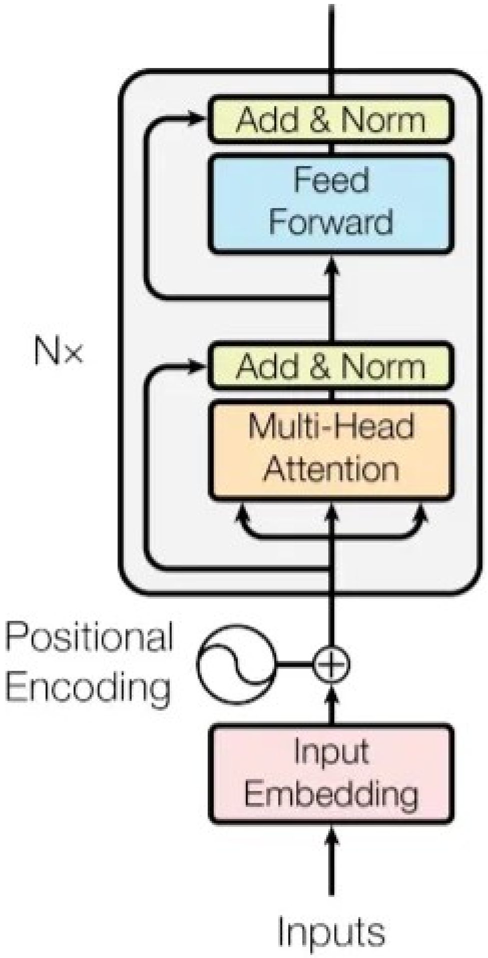
including personally identifiable information names, phone numbers, addresses, etc.,



How does it work?



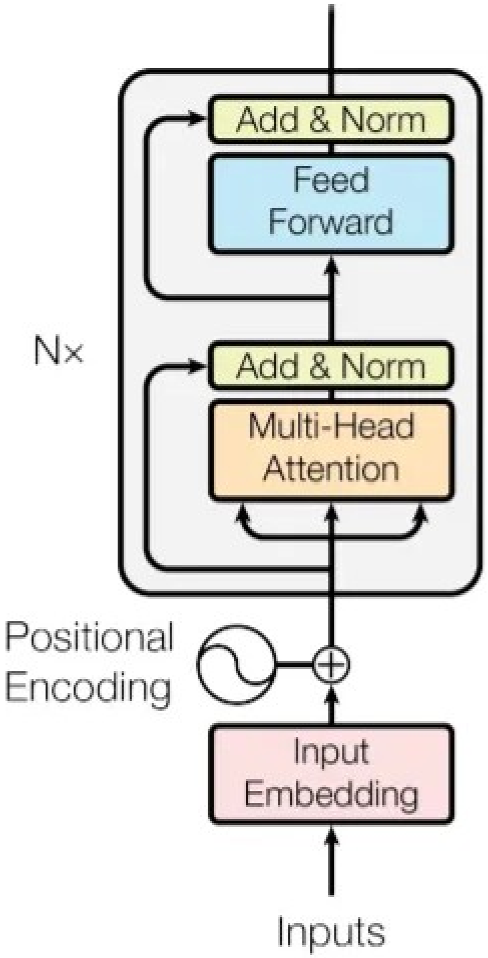
Step-by-step overview of the encoder part



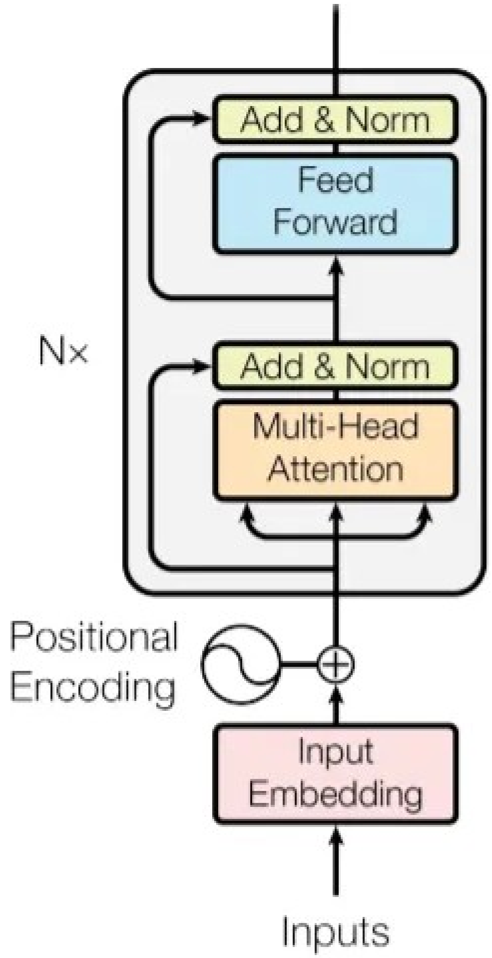
Step-by-step

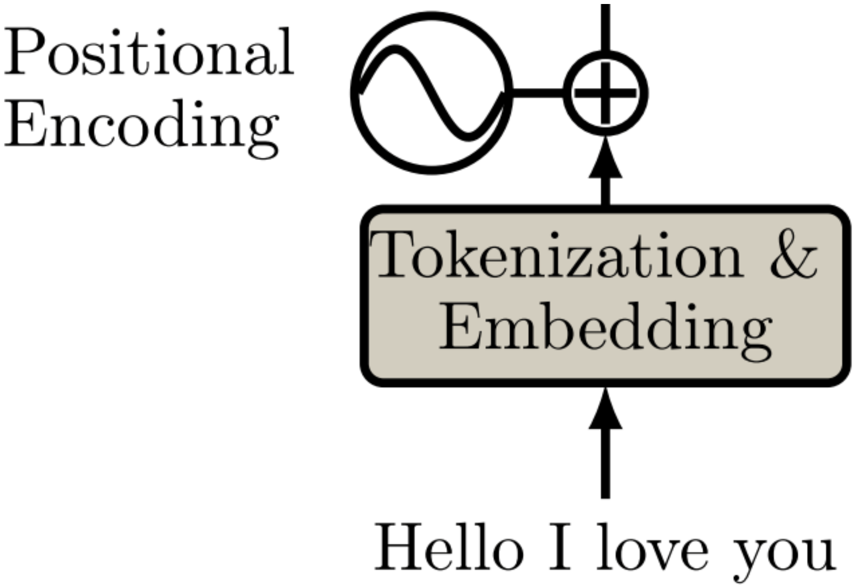
Transformer →

Pre-processing →



pre-processing





Input, variable size

Tokens represented as a 'set'

Text	Tokenization
	"Hello", "I", "love", "you"
"Hello I love you" ⇨	"love", "Hello", "I", "you"
	"I", "love", "Hello", "you"

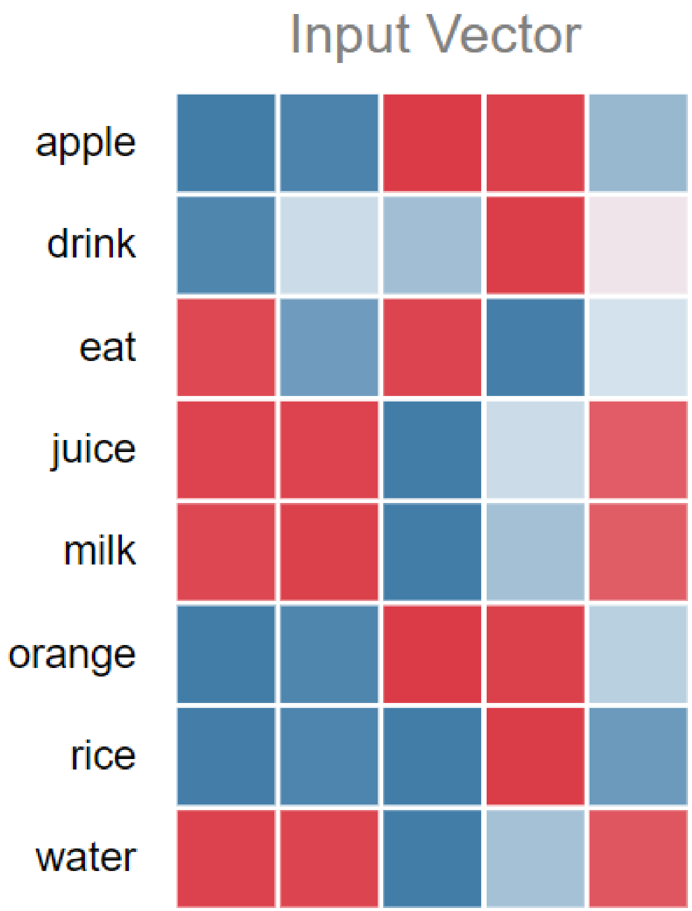
Word-embeddings

Words → vector embeddings (list of learned numbers)

Pretrained on large text corpa:

word2vec and Glove

Can also be learned by the transformer!



Similar words meanings, similar vectors

4.2 Vector similarities and analogies

If the created word vectors indeed capture some form of semantic meaning, words with similar meaning should be clustered together. A few examples can be seen in table 4.

PEPTIDE	TYR	COUGH	DISEASE	BACON
PEPTIDES	PHE	COUGHING	DISEASES	FRIED
N-TERMINAL	LYS	WHEEZING	PATHOGENESIS	SALAMI
N-TERMINUS	ASN	DYSPNEA	DIAGNOSIS	MEATS
POLYPEPTIDE	LEU	DYSPNOEA	CHRONIC	MEAT
C-TERMINAL	ARG	THROAT	CLINICAL	PORK
AMINO	ASP	SNEEZING	SEVERE	COOKED
EPITOPE	GLY	SORE	PATIENTS	GRILLED
SYNTHETIC	THR	BRONCHOSPASM	ETIOLOGY	BUTTER

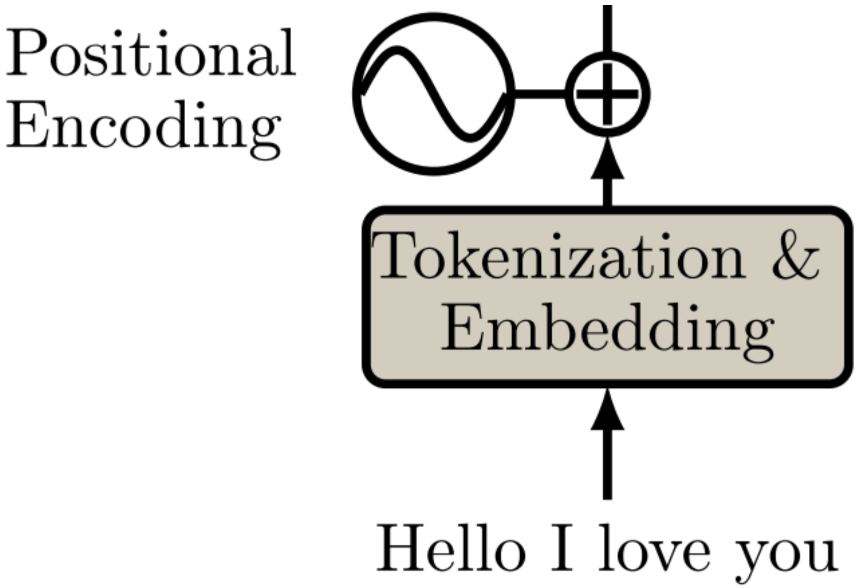
Table 4: Word embeddings closest (cosine distance) to query.

Wordvectors capture relations between words

Input A	Input B	Input C	Answer
MOUSE	MICE	STUDENT	**STUDENTS**
ESCHERICHIA	COLI	ARABIDOPSIS	**THALIANA**
COW	HAMBURGERS	PIG	**HOTDOGS**
BRAIN	NEUROLOGY	BLADDER	**UROLOGY**
LEUKEMIA	DASATINIB	MELANOMA	**SUNITINIB**

Table 5: word vector results (shown in bold) if the relation vector of the first two words is given to the third word.

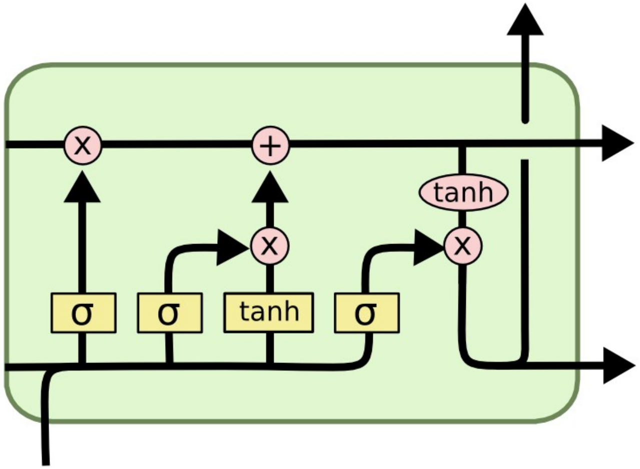
Positional encoding



Location information

- RNNs process information sequentially using an internal memory

- first input, memory update, second input memory update.... last input, final memory



Location information

- Transformers use ‘positional encoding’
- Given the location of word in the sentence produce a vector representing that location
- This using Sin and Cos functions

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/512}}\right)$$

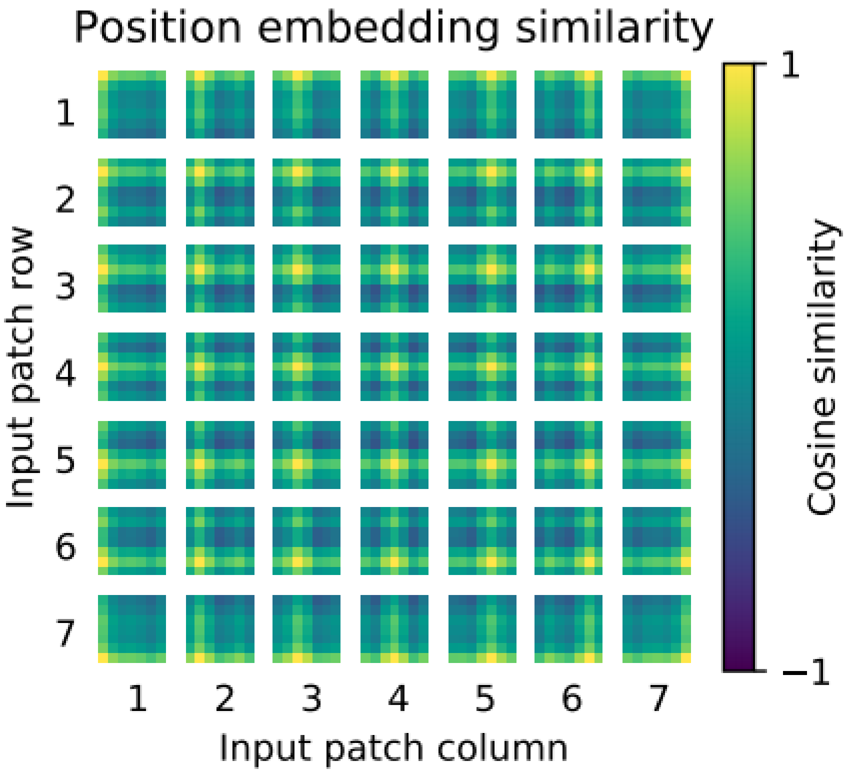
$$PE_{(pos,2*i+1)} = \cos\left(\frac{pos}{10000^{2i/512}}\right)$$

Location information

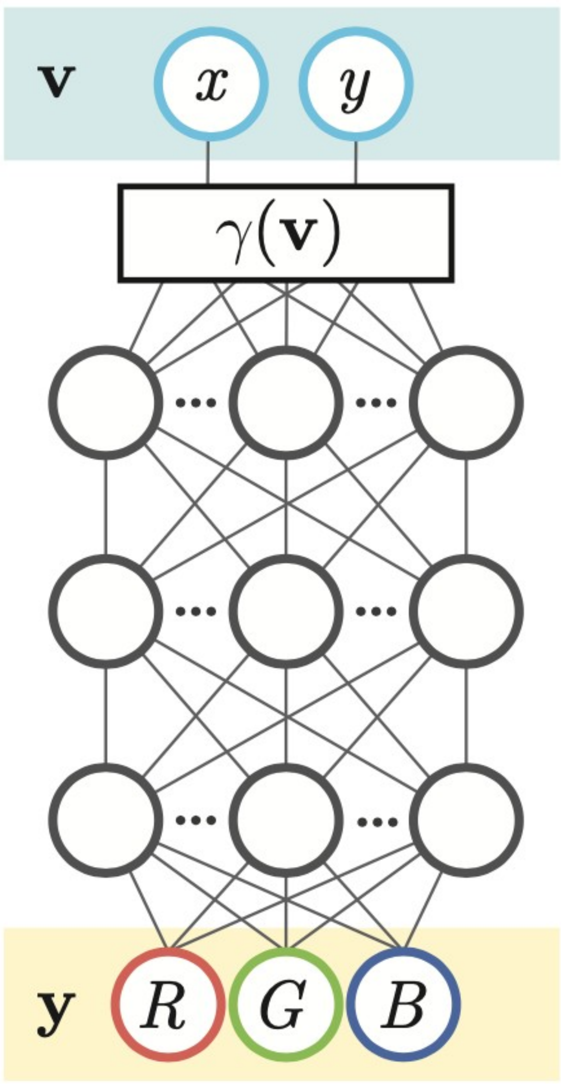
These ‘position embeddings’ can also be learned by the transformer

AN IMAGE IS WORTH 16X16 WORDS:
TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy*,†, Lucas Beyer*, Alexander Kolesnikov*, Dirk Weissenborn*, Xiaohua Zhai*, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby*,†
*equal technical contribution, †equal advising
Google Research, Brain Team
{adosovitskiy, neilhoulsby}@google.com



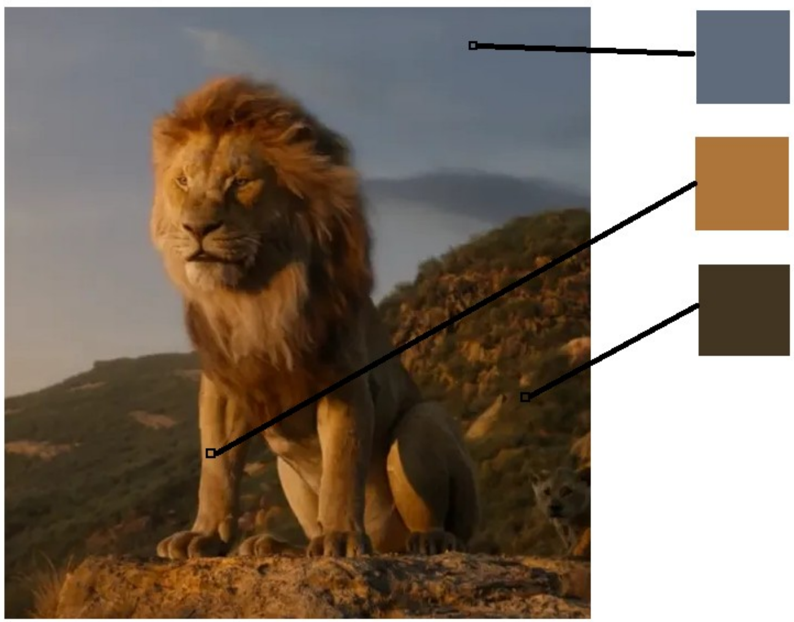
Neural networks have a "spectral bias" towards being smooth



Treat image as a 'function'

Input: the x,y coordinates (normalize between 0-1)

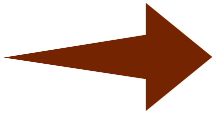
Output: the RGB value on that location



Example

If position is represented as 2 real values (0-1)

GT



output

Many many many neurons and parameters + weeks of training:

Example

If x and y positions are represented through random sin functions

GT

output

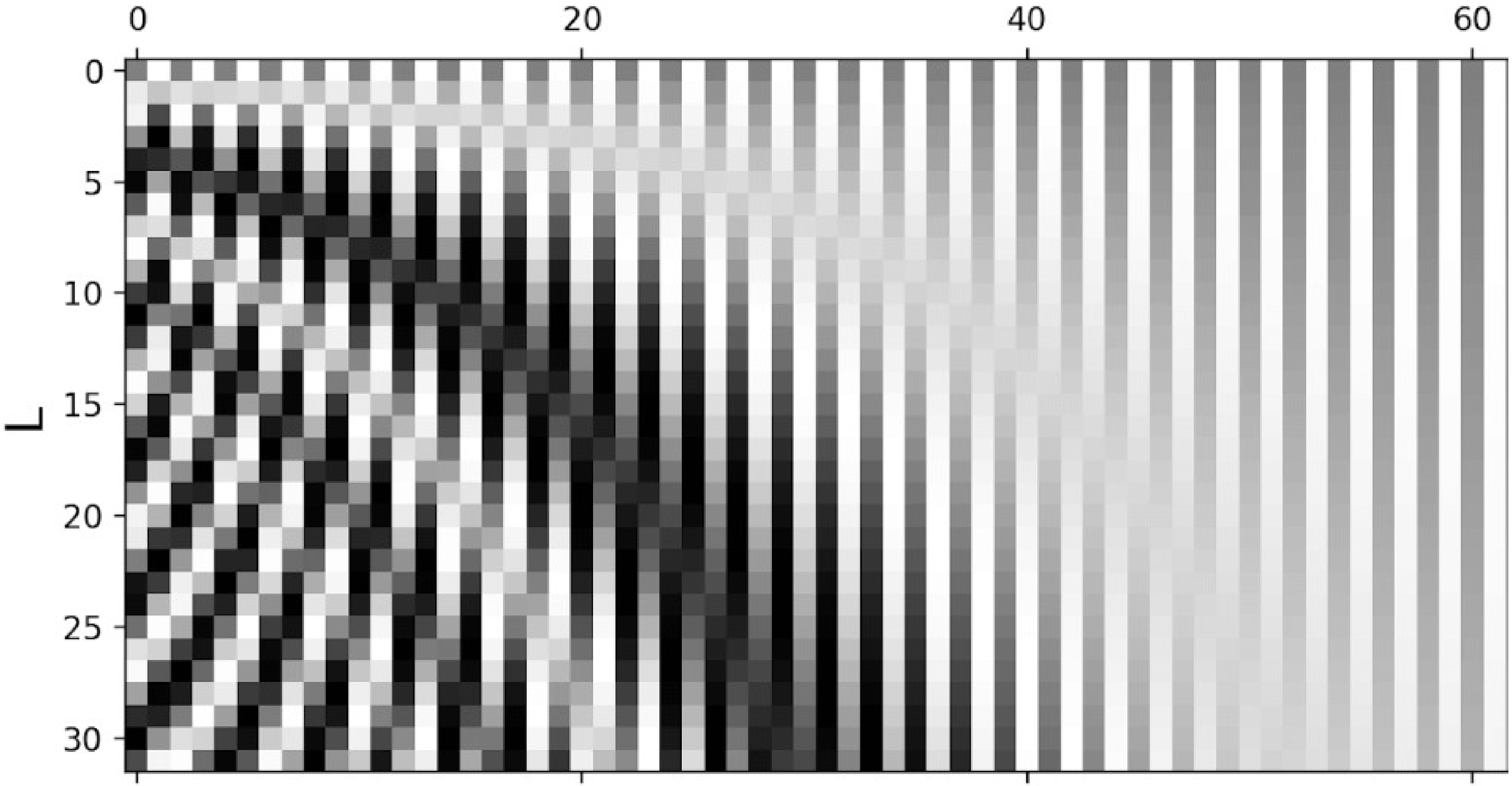
takes seconds to train and much much better results!

Formula

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/512}}\right)$$

$$PE_{(pos,2*i+1)} = \cos\left(\frac{pos}{10000^{2i/512}}\right)$$

Make it visual

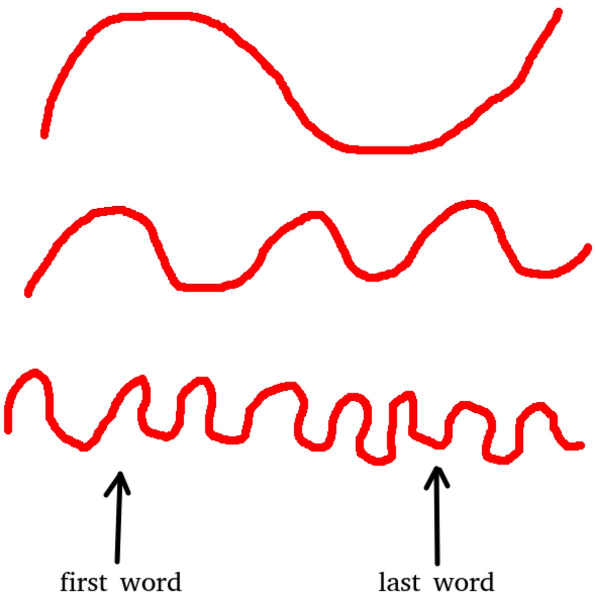


Visualization of the sinus function

https://lilianweng.github.io/lil-log/2020/04/07/the-transformer-family.html

Position encoding

Multiple wavelengths are used to give the vectors position information



Large wavelength:

rough estimation

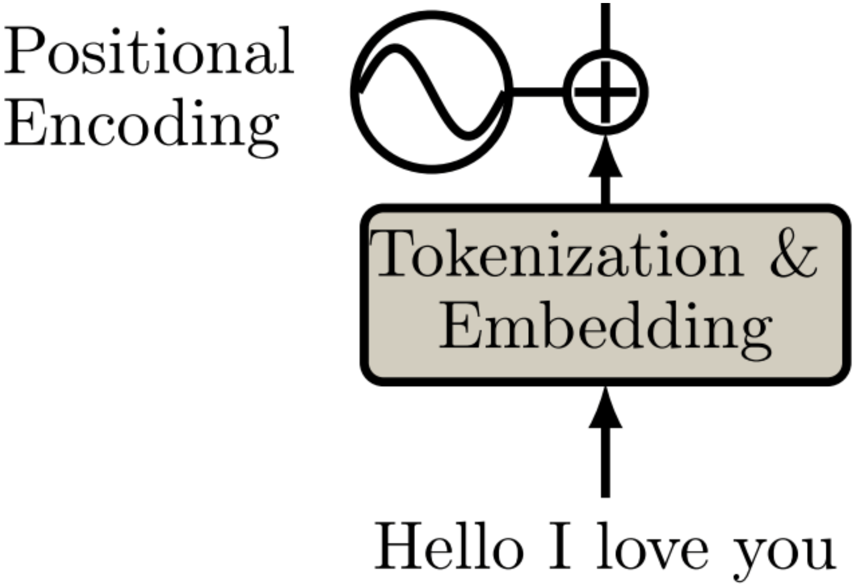
Small wavelength:

precise information

Positional encodings

Add the positional encodings to the word vectors

(not concatenated)

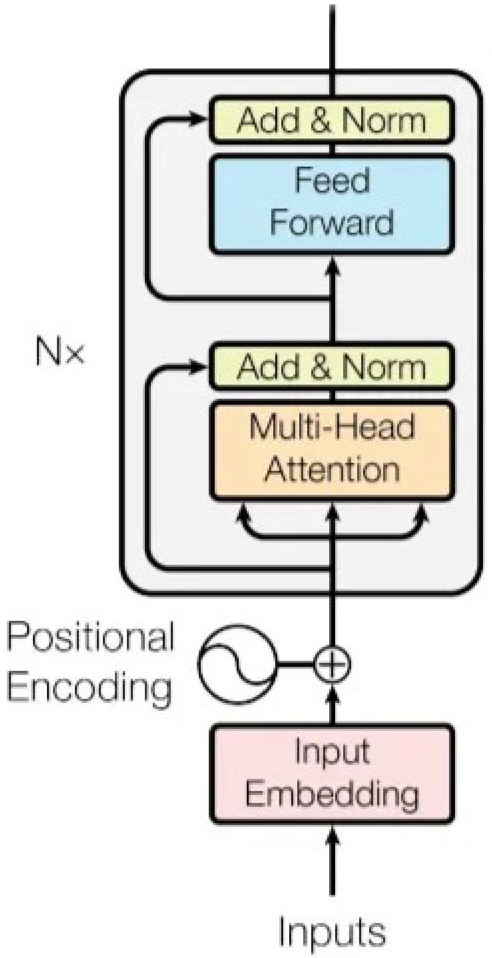


Next step

The actual transformer

This unit is often repeated multiple times

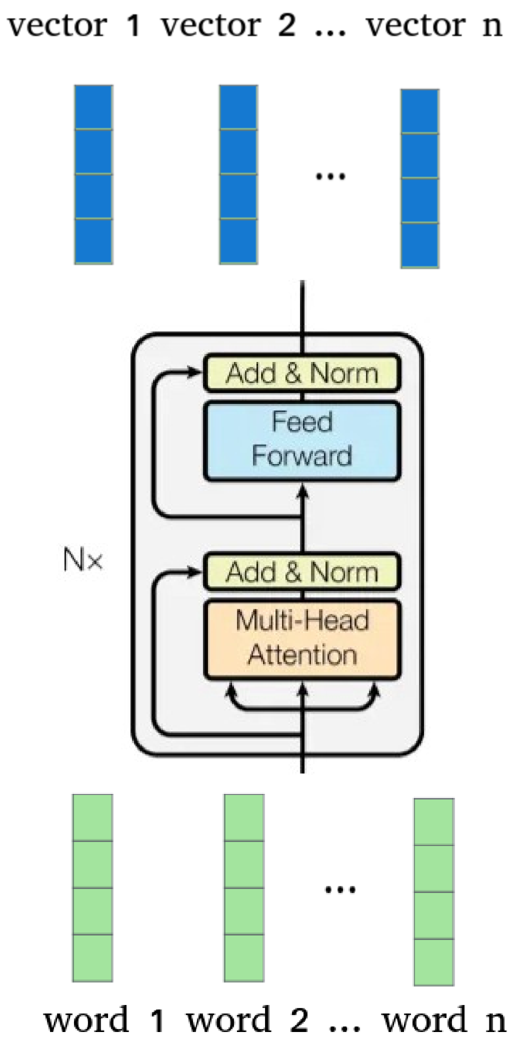
$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$



Input output

Before we go to the details

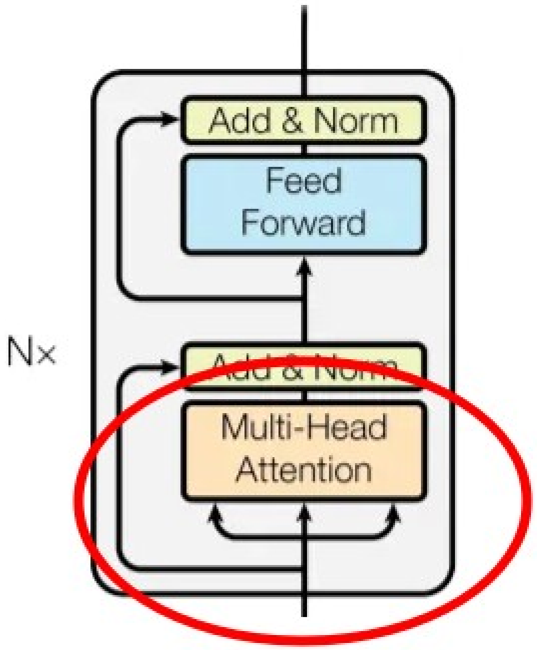
Same number of vectors come out as you put in with the same dimension!



Next step

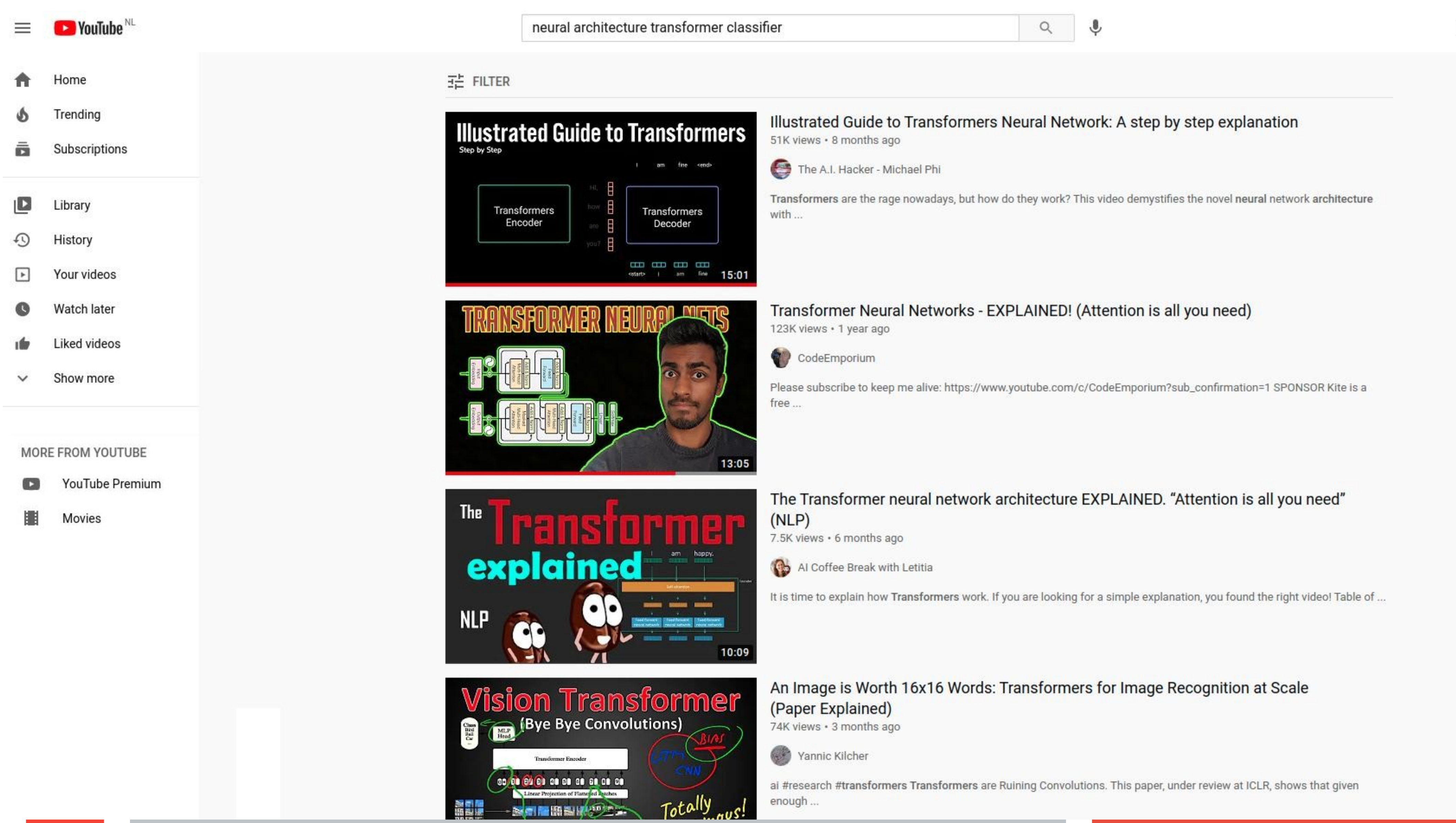
From the video we saw last week:
Key, query and value vectors were
created for each of the inputs

$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

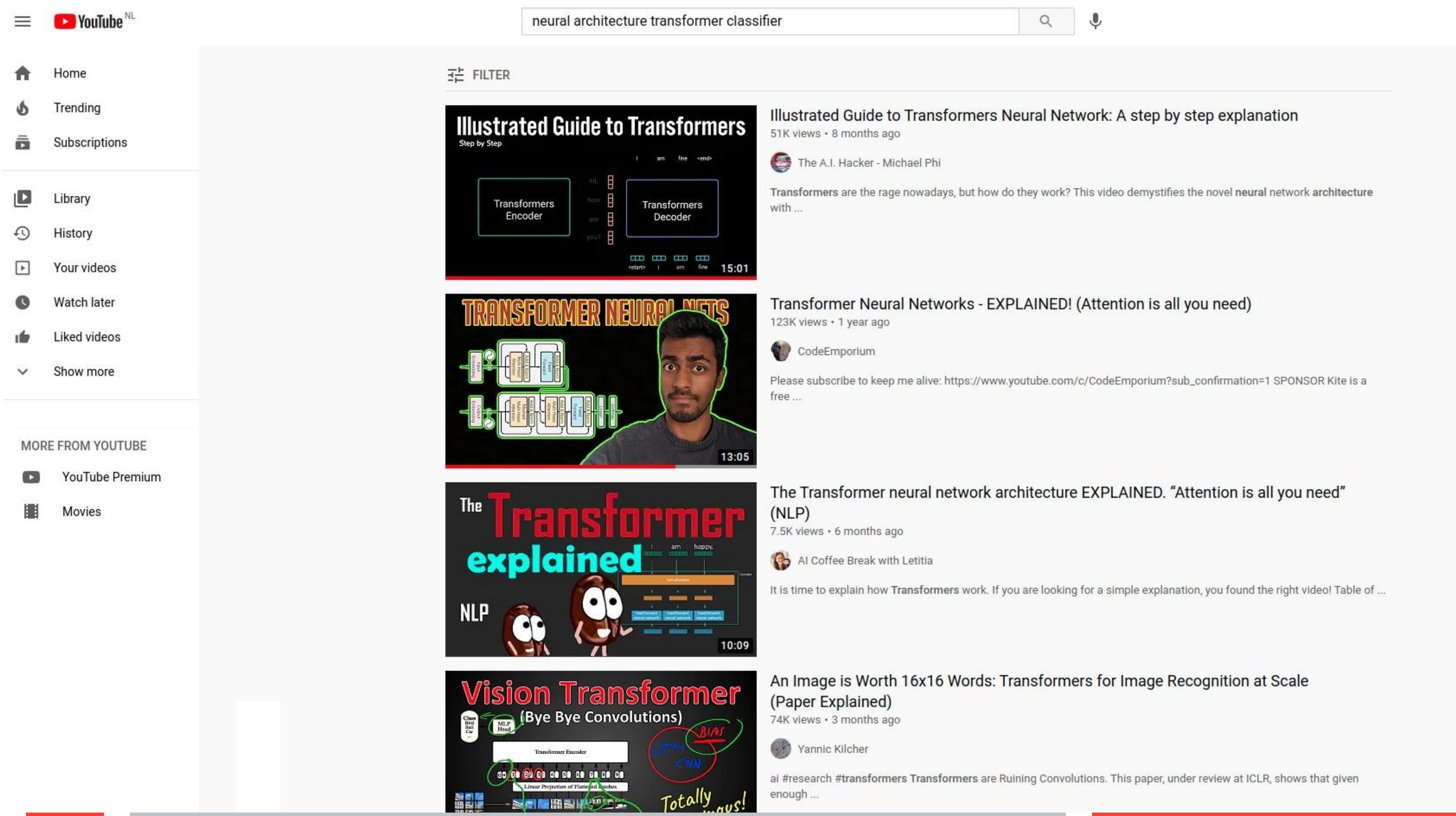


Lets make it more intuitive before we
go to the details

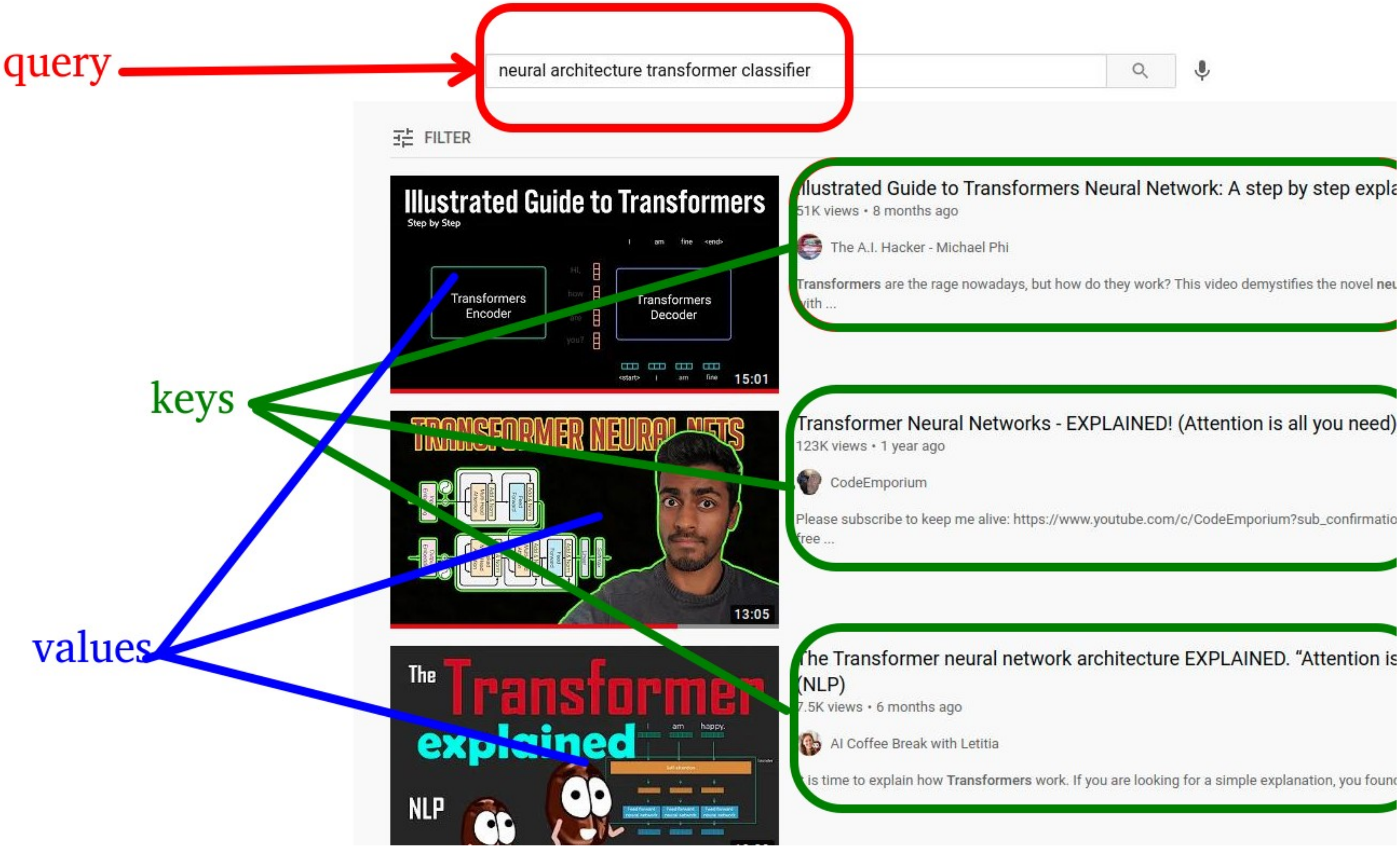
Keys queries values



Keys queries values



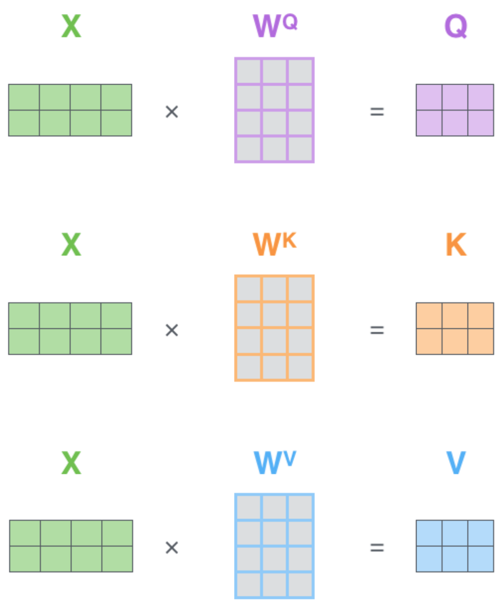
Keys queries values



Query key value

For each input **x** (each word-embedding) create **3** vectors
One **key** vector, one **query** vector and one **value** vector

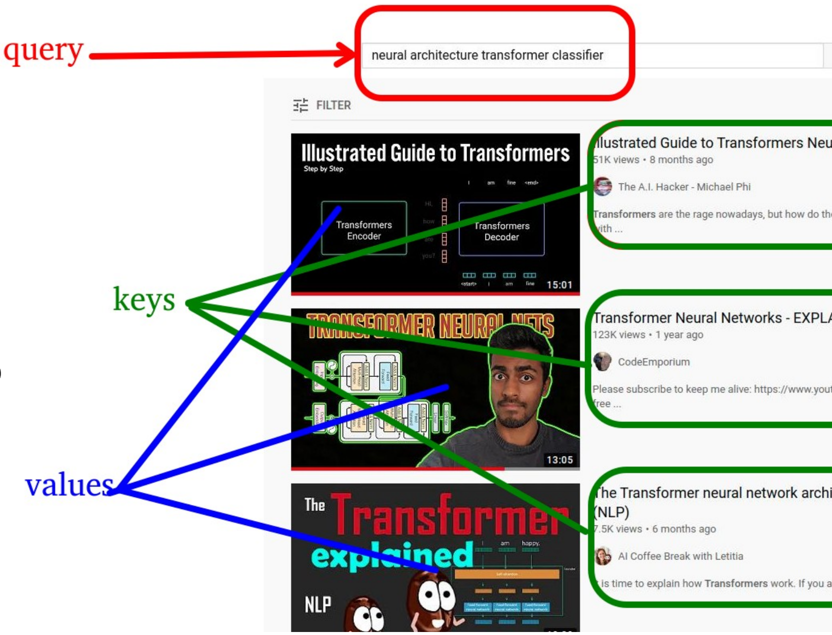
Can be done with simple **linear layer**
Mathematics to create these vectors is identical, only parameters differ



Q K V

Video score is based on the overlap between key-words and query-words

The more your query matches the keys the better the video score



How to do this with vectors?

K Q

Simple, use the dot-product between the key and query vectors!

Example:

$$\text{dot}\begin{pmatrix} 4, & -2, & 1, & 3 \\ 5, & -2, & 1, & 2 \end{pmatrix} = 31$$

$$\text{dot}\begin{pmatrix} 4, & -2, & 1, & 3 \\ 1, & 2, & -2, & 1 \end{pmatrix} = 1$$

$$a \cdot b = \sum_{i=1}^{n} a_i b_i$$

a = 1st vector

b = 2nd vector

n = dimension of the vector space

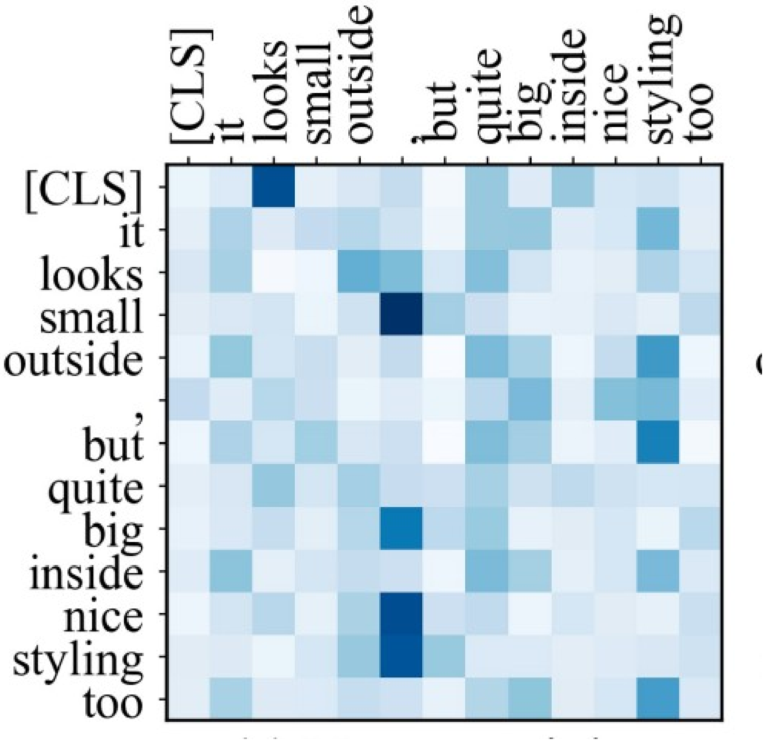
a_i = component of vector a

b_i = component of vector b

Attention matrix

Match **all key** vectors against **all query** vectors

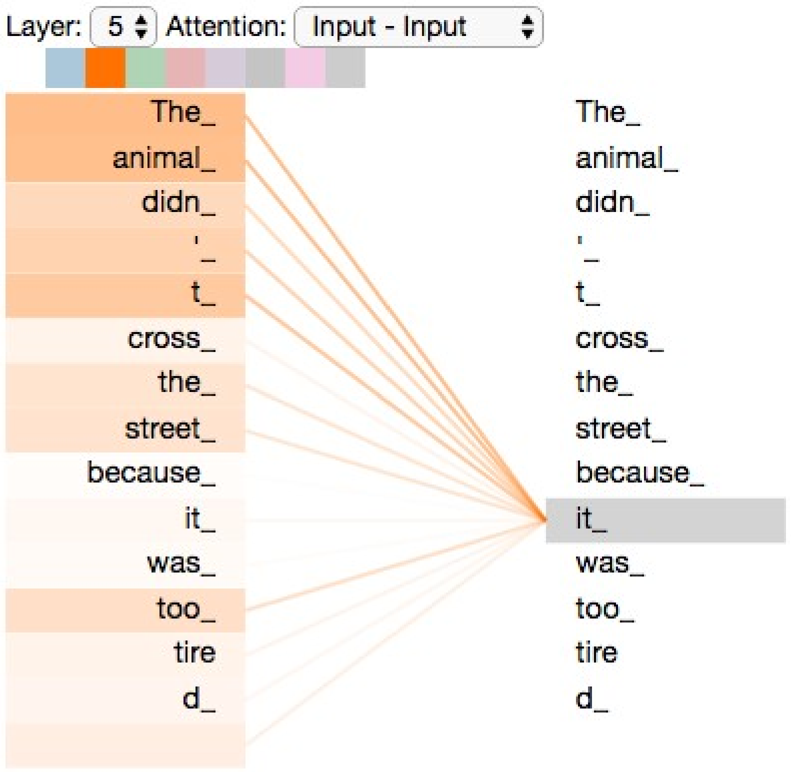
And use **softmax** to **normalize** the values between **0** and **1**



https://www.aclweb.org/anthology/D19-1361.pdf

Attention

For each word, how much attention



Take

Use this as ‘attention’ to the value-vectors

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

(root(dk) is to stabilize the gradients)

Multiheaded attention

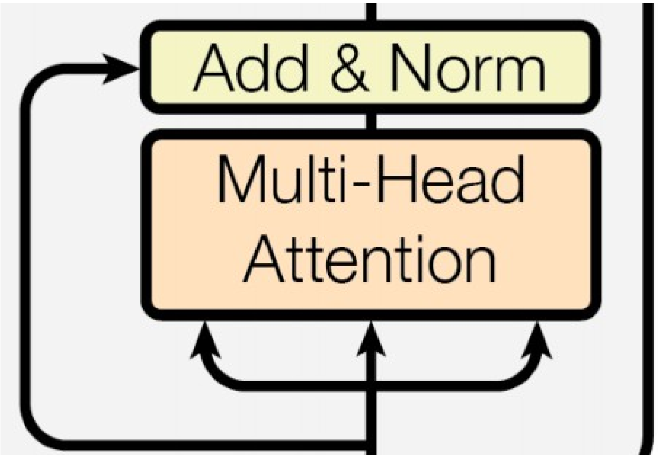
Multi head?

Instead of three big vectors for key query and values

Make multiple small ones and concatenate the results together

Each subpart can learn to focus on something different

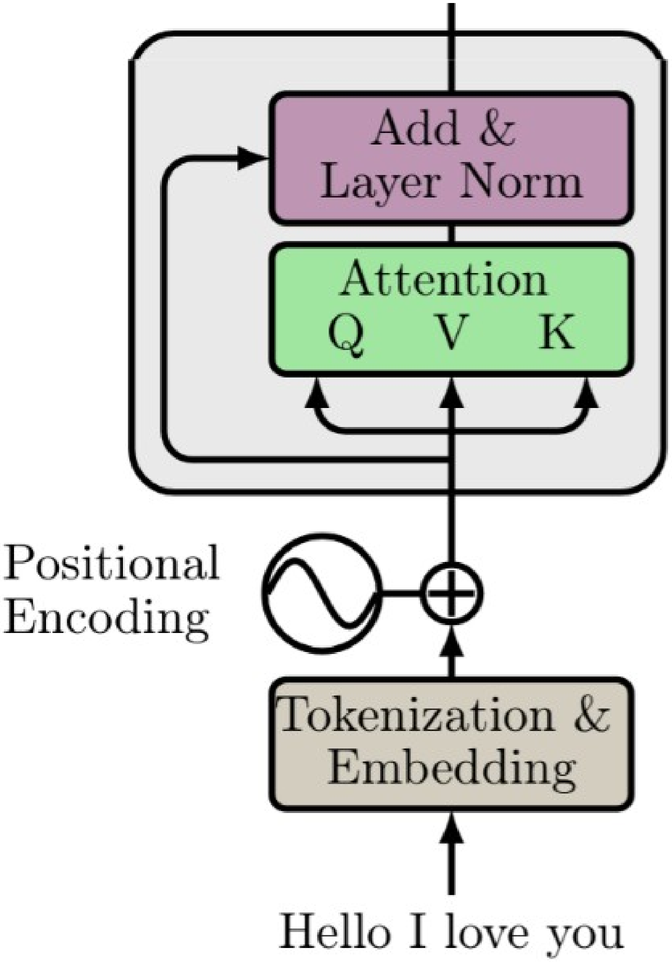
Each subpart has own softmax that sums to 1 which makes it harder to only focus on single value vector



Skip conncetion and normalization

Skip connections

Add (not concatenate) these new vectors to the original input vectors and normalize



Layer normalization

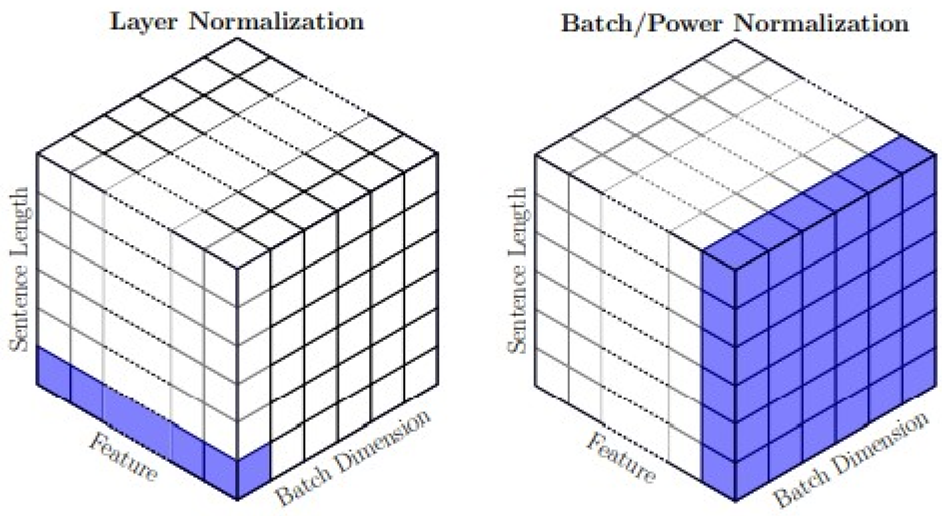
With batch-norm:
normalize over the batch

→ for each feature subtract mean and divide by std

With Layer-norm

Normalize over the layer

→ for each layer normalize all outputs (before activation)



Not dependent on batches, same output in training or evaluation mode

Linear layers

After attention part,

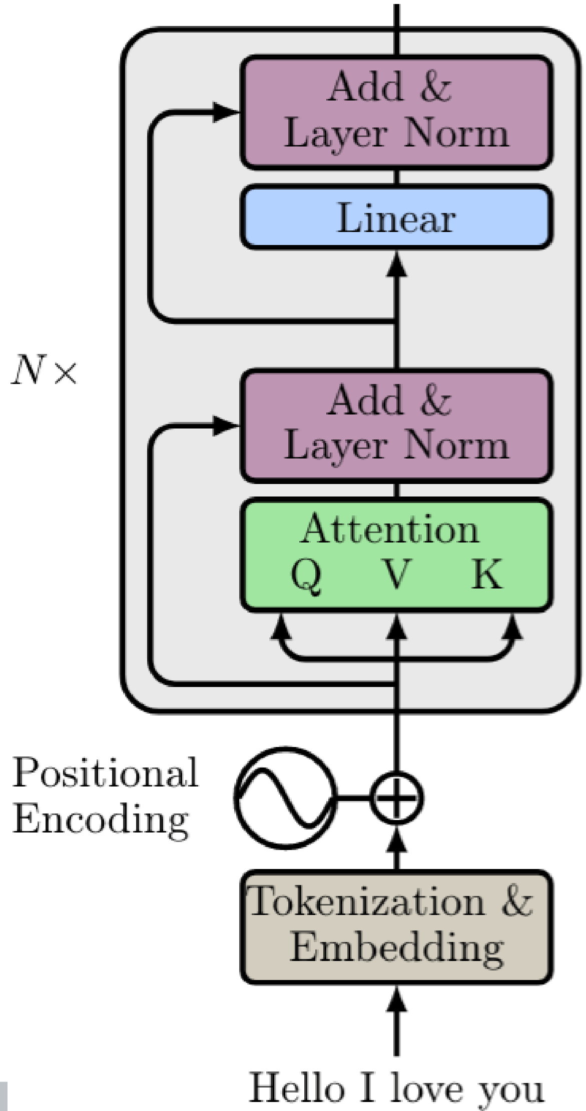
Linear layers part

Standard resnet architecture

Input → 2 or more layers →

skip connection → normalization

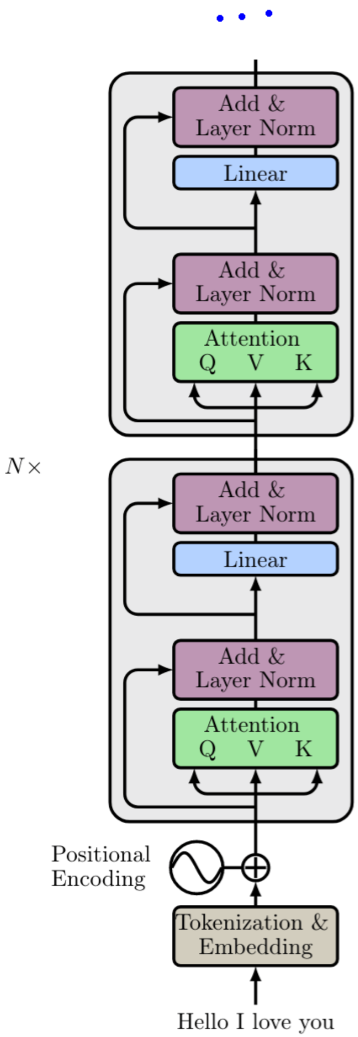
(no information sharing between vectors)



Stacking

Often multiple transformer blocks are used

Original paper used 6 of these blocks

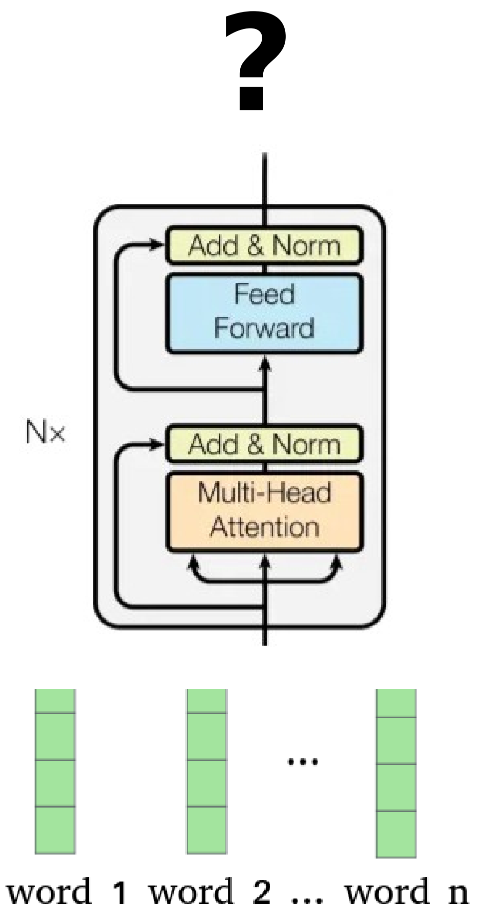


Classification

How many outputs?

How would you do classification?

(using this architecture)

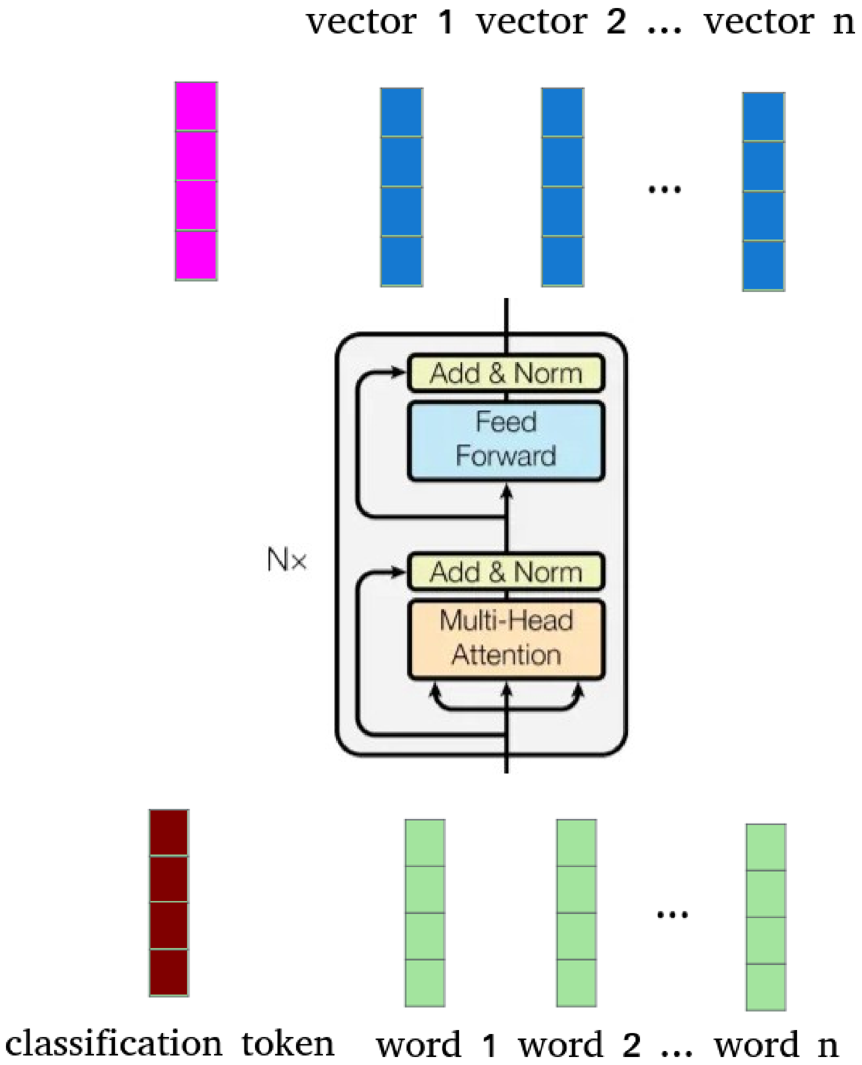


Classification

Add a ‘special classification token’

Take last representation of this token and use simple linear layer followed by softmax to get the classifier

(or without softmax for regression problems)



Next week

Generative transformers

Plus demo of how to code a transformer in pytorch